L5: Entry 2 of 2 File: USPT Sep 8, 1992

DOCUMENT-IDENTIFIER: US 5146576 A
TITLE: Managing high speed slow access channel to slow speed cyclic system data transfer

Abstract Text (1):
A cached DASD controller is illustrated which is attached to a high speed serial channel, such as an optical fiber channel. The data rate of the serial channel is much greater than the data rate of a DASD connected to the controller. The serial channel has a relatively long propagation time such that synchronous operations between the host processor 10 and the DASD cannot be efficiently performed. Operation of a data transfer, whether read or write between the host processor 10 and the DASD is monitored. Whenever reading a copy of the track contents stored in cache or the DASD reaches either an index mark, a break point from a roll mode operation or certain write operations occur resulting in predetermined data being stored in cache, then a GOCACHE flag is set in a control portion of the controller. The device operations are then momentarily idled while cache to host processor data transfer operations are enabled.

Brief Summary Text (4):
Direct Access Storage Devices (DASD) are a class of disk data storage devices currently used in data processing environments. Typically, the burst data rate between the DASD and its associated electronic circuits, such as a cache memory, varies from less than two megabytes per second through six or so megabytes per second. Because of such high burst rates and the electronic speeds of circuits attached to the DASD, it has been common practice to synchronize the operation of a host computer access to a DASD to the rotation disk being accessed. This mode of operation is called a synchronous mode, i.e., the operation of the data transfer is synchronized to the rotation of the disk in the DASD. Such synchronous operations are satisfactory up to a predetermined lengths of signal cables extending between the DASD and its controller and connecting host processors. The reason for the limitation in the spacing is propagation time of the signals between the host processor and the DASD controller. Such propagation time, if extended beyond a predetermined maximal elapsed time, requires a longer time to connect and reconnect to a host processor and exchange control signals then the DASD uses to scan a gap between a control field and a data field on the disk surface. If the propagation time exceeds the gap scanning time, then an additional rotation of the DASD disk is required for reading the data field after scanning a control field. That is, in CKD architecture (count, key, data) the count and key field always precedes the data field for each record. Except in a FORMAT WRITE operation, the count field is in a read mode while the data field can be either in the read or write mode. Using the CKD architecture, it is required to read the count field, scan a gap, and then read the data field. If the gap scanning time is less than the control signal propagation time, such synchronous operations are impossible. The above discussion does not describe all possibilities, the discussion is intended to explore a need for controlling and accessing DASD in other than a synchronous accessing mode.

Brief Summary Text (5):
With the advent of optical fiber channels having a data burst rate much greater than the data burst rate of the DASD, such as four times the burst rate. The data rate is combined with extra long cable length, such that a propagation delay is greater than the gap scanning time of the DASD, the rearrangement creates performance problems for a DASD peripheral subsystem. It is also desired to maximize utilization of the channel, i.e., maximize utilization of the optical fiber channel such that once a burst of data and its control signals are transferred over the channel, such transfer proceeds for as long as possible. Rate changing buffers can accommodate

some of the burst rate differences; however, more control is needed for efficiently using a DASD with such a fiber optic channel that has long propagation delays. It is desired to operate the DASD and the channel in a non-sychronous mode such that operations with both the channel and the device can be optimized to a maximal extent for the data transfer being effected between the host processor.

Brief Summary Text (7):
U.S. Pat. No. 4,912,630 shows batch or burst data transfers between slow and high speed systems. This patent teaches that data or signal bursts are limited by a elapsed time, i.e., a number of processor cycles. It is desired to avoid such an elapsed time limitation on bursts of data transfers.

Brief Summary Text (8):
U.S. Pat. No. 4,583,166 shows a cylindrical roll mode of DASD disk accessing and the use of a host channel type chain of commands within a peripheral subsystem. A so-called roll mode more efficiently uses a DASD in that data transfers can start at any rotational position of the disk. Such a roll mode is most effective for those data transfers involving one or more complete tracks of data to be transferred. The onset and termination of a data transfer beginning at any byte or record location on a track is called a break point. Such a break point is used as one logical criterion for controlling a peripheral subsystem in accordance with the present invention.

Brief Summary Text (9):
U.S. Pat. No. 4,214,742 shows an optical fiber serial channel to DASD connecting controller having a microprocessor. Several data paths, serial or parallel, are controlled by the microprocessor to effect different data transfers. The present application provides different data paths for different functions under microprocessor control, such as shown in this reference. In particular, a control of a rate-changing buffer and a cache are used in combination for maximizing data transfers between a DASD and the high data burst rate of a serial channel having long propagation time delays.

Brief Summary Text (13):
It is an object of the invention to provide an enhanced method and apparatus for transferring data between a cached DASD or other data processing subsystem and one or more host processors; using a high speed channel having a long propagation time.

Brief Summary Text (14):
According to the invention, a method of operating a cached peripheral controller which connects a high speed channel to a low speed device includes the steps of transferring data between a channel and the device via a controller, storing a copy of the data being transferred into a cache, monitoring the device operation for detecting that the data transfer has reached a predetermined status at the device; then terminating the data transfer from the channel, stopping the device activity as to the instant data transfer, and activating the cache and channel for transfers therebetween which do not involve the device. Such a decision is made independent of whether or not the data transfer between the host processor and device has been completed.

Brief Summary Text (15):
Apparatus using the present invention includes a peripheral subsystem having a plurality of peripheral devices, data transfer circuits connected to each of peripheral devices, a cache and a rate-changing data buffer being connected to the data transfer circuits for exchanging data signals therebetween, host attachment circuits connected to the buffer and said cache for exchanging data signals therewith and control means connected to the attachment circuits cache, buffer and data transfer circuits for controlling their respective operations including a GOCACHE flag which indicates that data transfers between one of the peripheral devices stops while data transfers between the cache and attachment means ensue. Control means monitor the operation of the device for determining the time to switch data transfer mode between device with attachment circuit operation to cache and attachment circuit data transfer operations.

Detailed Description Text (3):

Each of said DASD's 20 preferably has a capacity of 100 or more megabytes, which stores or caches data for a more rapid access by host processor 10, as is well known. Data transfer circuits 18 are those electronic circuits used to modulate the signals to be recorded on DASD 20, demodulate signals received from DASD 20, provide error detection and correction operations on such data, add and delete control characters and fields, as is known for such DASD operations. Operation of data transfer circuits 18 is synchronous with the rotation of a disk in DASD 20 which is currently being accessed. Each peripheral subsystem 19 generally will contain a plurality of DASD's 20 as indicated by ellipsis 21. Each of the data storage disks (not separately shown) have an index or reference circumferential position 20A.

Detailed Description Text (4):
Data bus 24 contains control and data lines for enabling attachment circuits 11 to communicate with rate changing buffer 13 and has an extension 25 to cache 16 such that signals from host processor 10 can be received both from data buffer 13 or cache 16. During a write or recording operation, the data signals can be supplied only to cache 16 or only to rate-changing buffer 13. The bus 22 connects rate-changing buffer 13 to data-transfer circuits 18 with a side bus-connection 23 to cache 16 such that signals from rate-changing buffer 13 can be recorded on DASD 20 and simultaneously copied to cache 16. Data read from DASD 20 is supplied to either or both the rate changing buffer 13 and cache 16. Control line 27 represents a plurality of electrical connectors extending between control and microprocessor 14, hereafter microprocessor, and other major elements 11, 13, 16 and 18 of the controller for subsystem 19.

Detailed Description Text (5):
Microprocessor 14 includes a dispatcher or an executive which manages the operation of the microprocessor for controlling the peripheral data subsystem, as is known and practiced. Some of the microcode modules are used exclusively for providing specific functions involving the data transfers. These modules will be described enable an understanding the practice of the present invention. Microcode module DVE PROC 30 shown logically, as being next to data transfer circuits 18, actually resides physically in the microprocessor 14, is that portion of the controller microprocessor 14 for managing the data transfer circuits 18 and for effecting the data transfer operations between DASD 20, rate changing buffer 13 and cache 16. These types of operations are well known and not further described for that reason. Similarly, CD PROC 31 is that microcode module of used by microprocessor 14 for controlling the operation of attachment circuits 11. Such controlled operation transfers data signals between host processor 10 and rate changing buffer 13. Under DVE PROC 30 control, in a "branching" write operation, a copy of the data in rate changing buffer 13 is supplied to DASD 20 and is also supplied over bus extension 23 for storage in cache 16. Microcode module CC PROC 32 of microprocessor 14 controls data transfers between host processor 10 and cache 16 which do not involve rate changing buffer 13 nor DASD 20. Transfers of signals between host processor 10 and a cache 16 is well known and not detailed for this reason. With each record 36 stored in rate changing buffer 13 is a record control field RCF 37. Referring to FIG. 2, RCF 37 is shown as having a CD PROC portion field 40 for use during writing operations which is controlled by CD PROC 31 and read by DVE PROC 30. RCF 37 is initialized by CD PROC 31 on a write operation and initialized by DVE PROC 30 on a read operation. Upon a read operation, flag bytes (not shown) are set by DVE PROC 30 to inform CD PROC 31 the requirements for the record being transferred and the number of records remaining in the track to be read. Since the DVE PROC 30 and CD PROC 31 are known and not necessary to an understanding of the invention, these two modules are not further described. Setting GOCACHE during a DASD read operation can be for a host command read at either a breakpoint or track index or after a write domain has ended and a DASD 20 read mode is established. In some instances several writing operations may occur in one host processor 10 defined write domain. GOCACHE could be set at the end of a one of the several writing operations which is within the host processor 10 defined write domain. GOCACHE byte 42 is a flag set by DVE PROC 30 when a DASD 20 portion of a data processing operation is a DASD read operation and reaches a predetermined status such that operations with the DASD 20 are to be interrupted and ensuing operations are preferably conducted with cache 16. Such a direct connection between host 10 and cache 16 enables a maximum data transfer rate over the optical fiber channel 12 while the data transfers with DASD 20 may be more efficient from a control view but are slower, i.e., do not maximize the burst rate of optical fiber channel 12. The operations conducted at the DASD 20 data transfer rate, and are interrupted or segmented using the GOCACHE flag, as will become apparent. Field 41 indicates the type of record contained in

the record image within rate changing buffer 13. This is used during read operations for enabling CD PROC 31 to know what type of record is being transferred of the CKD format. Four types of records are used in a constructed embodiment of the present invention. A first type of record is located at index or end of track, location. This first type of record in the illustrative embodiment is either a record zero, a home address record (the latter two are CKD control records) or a user record. Numeral 43 indicates that RCF 37 contains fields other than that shown in FIG. 2. The flow charts in FIGS. 4 et seq are directed toward operation of the illustrated FIG. 1 controller for a single device 20 which is currently transferring data with a host processor 10 via attachment circuits 11 and channel 12. It is to be understood that controller 19 simultaneously handles more than one device operation at a time. Therefore, a plurality of simultaneous device operations may occur, each of the device operations respectively relating to a different device 20.

Detailed Description Text (6):
Referring next to FIG. 3, an overall view of machine operations of the FIG. 1 illustrated data processing system is shown. Host processor 10 starts a data transfer operation with an address to one DASD 20 at machine step 50. The data transfer actually occurs between the host processor and the data processing subsystem at machine step 51. This step includes transferring data between the DASD 20, the host processor 10 and between the DASD 20 and cache 16 upon a read operation and between host processor 10 and cache 16 on a write operation. In machine step 52, DVE PROC 30 monitors the status of the data transfer operation with respect to DASD 20. The purpose of this monitoring is to determine a status in which the mode of operations that switch from device oriented operation to cache oriented operation for more effectively using the high speed channel 12. At machine step 53, DVE PROC 30 detects a predetermined end of DASD operations and sets GOCACHE flag 42 for indicating to the rest of the controller a host cache operation is to ensue. Machine step 54 represents that the cache-to-host mode is set which is indicated by the CD PROC 31, DVE PROC 30 being inactive, and CC PROC 32 being established for effecting the data transfers. Activating CC PROC 32 before it is needed reduces channel response time between the controller and the host processor 10.

Detailed Description Text (7):
FIG. 4 shows a read operation which uses the present invention for controlling the data transfers. A read operation involves asynchronous operations controlled by CD PROC 31 for transferring data from buffer 13 to host processor 10 and by DVE PROC 30 for transferring data from DASD 20 to buffer 13 and cache 16. The DVE PROC 30 operations are first described. A host command read operation is initiated by host processor 10 at machine step 60, i.e., the controller 19 may have received a read operation command and is starting DVE PROC 30 to requested data from DASD 20 to buffer 13 and cache 16. Such reading is also initiated by controller 19 for desired records. The operation is described for a host requested read. This read operation can be for one or more records stored on DASD 20. At each DVE PROC 30 controlled machine step 61 iteration, the controller 19 reads one of the host requested data records from a DASD 20 to buffer 13 and cache 16 for supplying same via buffer 13 as controlled by CD PROC 31 to the host processor 10. At the end of each record, DVE PROC 30 determines whether or not an index mark 20A on the track of FIG. 1 has been sensed. If an index mark 20A is sensed, then the scanning of the disk is at an end of a track being addressed, also called a current track. At machine step 63, DVE PROC 30 determines whether or not the next track is stored entirely in cache 16. Note that the next track accessed by host processor 10 can be the current track or another track. If yes, then at machine step 64 the GOCACHE flag 42 is set to the active condition for signaling to CD PROC 31 to send a channel command retry to host processor 10 and to transfer control 10 CC PROC 32. Operation of a channel command retry (CCR) is well known and not further described for this reason. Following setting GOCACHE flag 42, DVE PROC 30 stops device operations as indicated at machine step 65 and waits for another device operation to begin. If the entire contents of the next track being read was not stored in cache 16 at machine step 63, then machine operations return to step 61. As described in the Background of the Invention, a break point can be the beginning and end of a roll mode track read operation for DASD 20 or the location of a first record of a split track stored in cache. If the break point has been read, then the contents of the current track being read is in cache and steps 64 and 65 are then performed. However, if no break point has been reached, then the read operation should continue by repeating steps 61, et seq. The operation of steps 64 and 65 stops the device operation based upon reaching an index mark and finding the contents of the current

track and cache or reaching a break point during a read operation. If a next track is to be read and a copy of data stored in such next track is in cache 16, then the machine operations are then shifted from cache-to-host processor 10 operations for effecting higher data transfer rates enabled by the connection 12. It is to be understood that host processor 10 can terminate the FIG. 4 illustrated machine operations at any time, as is known.

Detailed Description Text (8):
The CD PROC 31 operations which occur asynchronously to the just-described DVE PROC 30 operations are next described. CD PROC 31 determines when a host processor 10 read command has resulted in records stored in buffer 13 are to be transferred over channel 12 to the host processor. The GET OP line 90 represents such determination. At machine step 91 a read operation of transferring data from buffer 13 to host processor 10. At machine step 92, CD PROC 31 checks the GOCACHE flag 42. If GOCACHE flag 42 is off, then data is to be transferred, i.e. such as one data record, from buffer 13 to host processor 13. This data transfer occurs at machine step 93. Following this data transfer, CD PROC 31 gets identification of the next operation to be performed, if the next operation is a host processor commanded read, then steps 91 et seq are repeated, otherwise operations beyond the scope of the present description are performed. If at step 91, the GOCACHE flag 42 is ON, then DVE PROC 30 has indicated that device operations should stop and cache operations should be initiated. At machine step 95, CD PROC 31 and DVE PROC 30 activities are stopped (note that reading of DASD 20 has already stopped and this action deactivates the DVE PROC module) and CC PROC 32 is activated for initiating cache 16 to host processor 10 data transfers. CD PROC is exited at 96.

Detailed Description Text (9):
A write operation using the present invention is illustrated in FIG. 5. At machine step 70, host processor 10 issues a "LOCATE RECORD" command. This command is a known command which sets an address space or write domain of DASD 20 address space for limiting the write operation. A write command follows the LOCATE RECORD command at machine step 71. The data transfers indicated by the write command 71, which can be for more than one CKD record, is effected at machine step 72. At machine step 73, whether or not the end of the write domain has been reached is checked. If not, an additional write command can be received at step 71. This description assumes that the issued write commands are such that there is no overrun in rate changing buffer 13 and that cache 16 has sufficient data storage space for all of the data to be written to DASD 20. In the event of a possible overrun, a control mechanism beyond the description of the present invention can issue a channel command retry (CCR) which interrupts the transfer of data signals from host processor 10 to the illustrated data processing subsystem to prevent overrun or to recover from an overrun error. Such overrun prevention and recovery is not pertinent to an understanding of the present invention. If the end of the write domain has been reached, i.e., writing is completed, then at machine step 74 a check is made whether or not a cache hit was made, i.e., was there data in the cache relating to the write command, if not, device read mode operations are entered at step 75. If a cache hit has occurred, i.e., data was in the cache 16 that was overwritten by the write command, at machine step 72, then at machine step 76 whether or not a copy of the entire track is in cache 16 is checked. If yes, if an entire track has been written and it is in cache, then the GOCACHE flag is set at step 77. On the other hand, if the entire track is not in cache, then at machine step 78, whether or not the HA record is valid is checked, i.e., was index passed, and the home address record has been stored in cache 16. If yes then at machine step 78 device read mode operations of step 75 are entered.

CLAIMS:

1. In a machine-effected method of operating a peripheral cached data storage subsystem, including the machine-executed steps of:

in a data signal transfer, transferring data signals between a connected host processor, a cache and a device in the peripheral cached data storage subsystem;

establishing a predetermined reference state of device operation that represents either scanning an index or a brake point of a current track being scanned;

monitoring the data signal transfer including monitoring operation of the device and the cache for indicating when the operation of the device has reached said predetermined reference state and the cache is storing a complete copy of a track of data signals that is being transferred in said data signal transfer, whether or not said data signal transfer has been completed;

detecting and indicating that the operation of the device has reached said predetermined reference state and that the cache is storing said complete copy of said data signals read from said current track being scanned; and

in response to said indication, stopping further operations of the device irrespective of whether or not the data signal transfer is complete and actuating the peripheral cached data storage subsystem for initiating said transfer of data signals between the cache and the host processor without accessing the device during said initiated transfer of data signals between the cache and host processor.

7. In apparatus for a peripheral data processing system, including, in combination:

a peripheral data processing device having a plurality of addressable data storage areas, each of said addressable data storage areas for storing one or more data records;

a cache connected to the peripheral data processing device;

a data buffer connected to the cache and to the peripheral data processing device;

attachment means connected to the cache and to the data buffer;

control means connected to the cache, data buffer, attachment means and to the peripheral data processing device;

data transfer means for transferring data records between the attachment means, the cache and the peripheral data processing device, including accessing a predetermined one of said addressable data storage areas as a current track being scanned, an index or a break point in said current track being scanned;

first means in the control means for monitoring the data record transfer and the operation of the device for detecting either said index or break point of said current track being scanned for indicating when the operation of the device has reached said predetermined reference state;

second means for monitoring the cache and the peripheral data processing device for detecting that a complete copy of data stored in the current area of the device is stored in the cache; and

said first means having third means responsive to said first means detecting that the operation of the peripheral data processing device has reached said predetermined reference state and to said second means detecting that a complete copy of data records stored in the current area is stored in the cache to indicate a gocache condition, fourth means connected to the third means for responding to the gocache indicating condition to stop further operations of the device irrespective of whether or not the data record transfer has been completed; said fourth means further responding to said gocache indicated condition for initiating operation of the cache with the host processor for transferring said data records between the cache and the host processor.

8. In the apparatus set forth in claim 7 further including, in combination:

register means for storing control flags including a GOCACHE flag and being connected to said third means and said control means, said third means setting said GOCACHE flag to the active condition for storing said gocache indicated condition; and

flag means in the control means for actuating the data transfer means for transferring data

records between the cache and the attachment means and being connected to the register means for sensing the GOCACHE flag, said flag means being responsive to the GOCACHE flag being set to the active condition for actuating the data transfer means to transfer data records between the cache and the attachment means without accessing either said peripheral data processing device nor said data buffer.

13. In apparatus for being connected to a host processor, including, in combination:

a cache and a rate-changing buffer;

a plurality of peripheral data storage devices, each of said device having a plurality of addressable data-storing portions, each of said addressable data-storing portions having a first type of record that indicates onset of a current data transfer between said cache and the respective device;

data transfer circuits connected to each of the peripheral devices;

said cache and said rate-changing data buffer being connected to the data transfer circuits for exchanging data signals therebetween;

host attachment circuits connected to said buffer and said cache for exchanging data signals therebetween;

control means connected to the attachment circuits, cache, buffer and data transfer circuit for controlling their respective operations and including a GOCACHE flag which indicates that data transfers between a one of the peripheral devices is to cease and data transfers between the cache and the attachment means may ensue; and

CC processor means in the control means for effecting data transfers between the cache and the attachment means, DVE processor means in the control means for effecting data transfers between any one of the peripheral devices with said buffer, said DVE processor means having means for setting said GOCACHE flag for indicating a predetermined state of operation of the peripheral device that indicates accessing a predetermined access point of an addressable portion of the device that is being currently accessed for transferring data signals to said buffer and means for sensing the GOCACHE flag for activating the CC processor means to establish a data transfer mode between the cache and the attachment circuits.

Previous Doc Next Doc Go to Doc#

Refine Search

Your wildcard search against 10000 terms has yielded the results below.

Your result set for the last L# is incomplete.

The probable cause is use of unlimited truncation. Revise your search strategy to use limited truncation.

Search Results -

Term	Documents
INCREAS$	0
INCREAS	19431
INCREASA	119
INCREASABILITY	12
INCREASABL	1
INCREASABLE	796
INCREASABLE-DECREASABLE	2
INCREASABLE/DECREASABLE	5
INCREASABLE/REDUCIBLE	2
INCREASABLY	14
INCREASABOVE	1
(L5 AND (INCREAS$ OR INCREMENT$)).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	0

There are more results than shown above. Click here to view the entire set.

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L6

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Saturday, February 25, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L6	l5 and (increas$ or increment$)	0	L6
L5	L4 and (track$ near (data near transfer$))	2	L5
L4	L3 and (data near transfer near system)	64	L4
L3	L2 and (manag$ near (data near transfer$))	386	L3
L2	L1 and ((transfer$ near data) same (host near computer))	10238	L2
L1	(transfer$ near data) and (host near computer)	25695	L1

END OF SEARCH HISTORY

L5: Entry 1 of 2 File: USPT Dec 3, 1996

DOCUMENT-IDENTIFIER: US 5581790 A
TITLE: Data feeder control system for performing data integrity check while transferring predetermined number of blocks with variable bytes through a selected one of many channels

Abstract Text (1):
Multiple numbers of "sets" of sender-receiver units operate concurrently to transfer blocks of data. The number of blocks to be transferred in each set is predetermined by a main host computer which registers the number-of-blocks-to-be-transferred into a protocol-controller in each set of sender-receiver units. An associated data feeder control system monitors the number of data blocks residing in a buffer memory, which has dedicated storage for each sender-receiver unit, and will only permit data block transfer to receiver units only to the amount presently available in the buffer memory until, eventually, the predetermined number of data blocks, for each set, is transferred to completion.

Brief Summary Text (2):
This disclosure relates to digital data transfer systems wherein means must be provided for the control and counting of the amount of data transferred between digital modules.

Brief Summary Text (6):
A basic operational function of systems involving digital modules has to do with transfer of data between various of the digital modules. Problems arise in regard to the accounting for and making sure that a specified amount of data has been transferred and has been transferred on a reliable basis with integrity.

Brief Summary Text (7):
As seen in FIG. 2A, there is indicated a block diagram of a data transfer system whereby data from a host processor 4 may be transferred to a channel interface module (CIM) 8.sub.c and then transferred to a device interface module (DIM) 8.sub.d from which the data can then be moved to one or more peripheral units such as the shown disk drive units 70. FIG. 2B indicates the DIM 8.sub.d with a protocol bus 6 connecting a buffer 24.sub.d to a Protocol Controller 80 with an integrity circuit 81 (Ic).

Brief Summary Text (10):
Subsequently, as seen in FIG. 3B, there may be data block transfers of a different volume or size as shown by the block designated "Y" bytes. Here again the header will provide the address destination, commands and other pertinent data after which the main body holds the data to be transferred followed by the EDC (error detection code) signature which characterizes the data being transferred.

Brief Summary Text (11):
When a block of data such as that of "X" bytes or "Y" bytes is being transferred, it is necessary not only that some means be provided to recognize the block size and to insure the integrity of the data transfer but that the entire specified number of bytes in each block and the number of specified blocks, has been transferred. As seen in FIG. 4, there is shown an example of two different block sizes which are to be transmitted between a sending and a receiving digital module, the first block size being of 180 bytes and having header and EDC signature. This is followed by a larger block size of 512 bytes which also has its own personal header and EDC signature.

Brief Summary Text (12):

The presently described system operates to insure that there will be no delay in the data transfer operation even though blocks of different sizes are being transferred and additionally, that the different block sizes will concurrently and on-the-fly be checked for integrity without any delay to the data transfer operation and that the specified proper number of blocks be transferred.

Brief Summary Text (13):
As seen in FIGS. 2B, 2C, the device interface module 8.sub.d (DIM) which is used to carry data to peripheral units such as a disk drive unit 70, will be seen to have certain functioning modules which enhance the transfer of data to the peripherals while at the same time providing for reliability in the data transfer operation by checking the integrity via I.sub.c, 81, FIG. 2B, of the data being transferred. As seen in FIGS. 2B, 2C data which has been passed down from the channel interface module 8.sub.c (CIM) and which temporarily resides in the memory buffer 24.sub.d, will then be passed on bus 6 to SCSI Protocol Controller SPC 80, which will provide the management for transferring the data on bus 78 to a selected group of disk drives 70. The drives 70 may represent a multiple number of peripheral units which can receive blocks of data or send blocks of data.

Brief Summary Text (14):
As seen in FIG. 2B, the bus 6 between the memory buffer 24.sub.d and the protocol controller SPC 80 is also connected to an integrity circuit I.sub.c 81. The integrity circuit 81 functions on-the-fly in order to check the integrity of the data being transferred. A control processor 10.sub.d provides initiation signals to the integrity circuit 81 so that it may provide its error checking function.

Brief Summary Text (15):
As illustrated in FIGS. 3A, 3B, blocks of data to be transferred can occur in different block sizes. FIG. 4 illustrates where each header of a block contains information as to the size of the block by denoting the number of bytes involved in the block. Then, after the pertinent data in the block is transmitted, the final portion of each block is seen to have an Error Detection Code (EDC) signature. This signature involves a Hexadecimal Code of, for example, nine bytes (36 bits) of which the first byte is a parity value.

Brief Summary Text (16):
Thus, in addition to checking the integrity of the data being transferred, there is also the problem of making sure that the entire block of data of, for example, 180 bytes or, for example, 512 bytes, has been transferred from one digital module to the other destined receiving digital module. The data feeder control F.sub.0, FIG. 1A, (DFC) of the presently described system performs the function of making sure that only the valid blocks of data available in the memory buffer 24.sub.d are transferred with integrity to the receiving module even though the host processor 4 has commanded a specific protocol-controller SPC 80 to transfer a larger total of "B" data blocks in total. In the case of a EDC error or a parity error, or in the case of all the bytes having been properly transferred from a sending module to the receiving module, then in that case, there is no more data available in the memory buffer for transfer. In each of these cases, the data feeder control (DFC) system will force a control logic unit so as to deny any further grants of protocol bus 6 to the protocol controller SPC 80.

Brief Summary Text (19):
In the present embodiment, data is transferred between a memory buffer, for example, as a first sending module over a channel bus to a receiving module designated as a SCSI Protocol Controller, SPC 80 which then transfers the data to a group of peripheral units, such as disk drives 70.

Brief Summary Text (20):
During the course of the data transfer, an integrity circuit (81) checks each word transferred for its parity value and also checks the entire block with an error detection code to make sure the integrity of the data block transferred has been properly fulfilled. Should an error in the parity of a given word occur or should an entire block be finally transferred with any EDC signature error, then the data transmission is interrupted by means of denying any further bus

grants for data transfer operations on that set of sender-receiver modules.

Brief Summary Text (21):
During the time that the various data blocks are being used to transfer data words and data bytes are moving from one sending module to a receiving module, at the same time there is provided a data feeder control (DFC) system which keeps track of the number of blocks which are properly transferred between the first sending module and the second receiving module, and the amount of data remaining to be transferred.

Brief Summary Text (22):
A device interface module (DIM) control processor 10.sub.d, FIG. 1A, is connected to a processor bus 12 which enables the processor 10.sub.d to communicate through an input register-buffer 14, FIG. 5, and through a I/O buffer 36, FIG. 1A, to a multiplexor 20 which is connected to a flag register 24. The multiplexor 20, FIG. 1A, also receives outputs from an interrupt register 34, a mask register 36 and a counter feeder control unit 32 which keeps track of the number of blocks of data that are being transferred between the sending digital module and the receiving digital module over on the protocol bus 6.

Drawing Description Text (2):
FIG. 1A is a drawing showing an integrity data transfer system which is monitored by a data feed controller;

Drawing Description Text (4):
FIG. 2A is a drawing showing the overall network of operations between a host computer through a channel interface module to a device interface module over to disk drive peripheral devices;

Drawing Description Text (6):
FIG. 2C is a more detailed diagram of the overall system whereby multiple numbers of host computers can communicate with multiple numbers of channel interface modules (CIM) connecting dual system busses which feed to multiple numbers of device interface modules (DIM) in order to provide data communications to and from multiple numbers of peripheral devices;

Drawing Description Text (10):
FIG. 6 is a timing diagram showing the clock cycles and functions performed during data transfer of a specified data block on a channel bus.

Detailed Description Text (4):
In conjunction with the data transfer between the first and second digital modules together with the integrity checking circuitry, the data feeder control (DFC) system provides a method for measuring the number of bytes and number of blocks of data which are to be transferred.

Detailed Description Text (9):
Outputs from the memory buffer 24.sub.d, the interrupt register 34 and the mask register 36.sub.m are also conveyed to the multiplexor 20 and also to the multiplexor 18. In this fashion, the output of the flag register 24, the output of the interrupt register 24, and output of the interrupt register 34 and the output of the masking register 36.sub.m can be conveyed through the multiplexors 20 and 18 to the processor bus 12 and to the protocol channel bus 6 in order to stop data transfer operations on those busses, by removing the bus grant.

Detailed Description Text (11):
The control processor 10.sub.d through its processor bus 12 can communicate with the protocol controller channel bus 6 or multiple numbers of such protocol controller channel busses such as may occur in a device interface module (DIM) which may have up to four such protocol control busses, 78.sub.0, 78.sub.1, 78.sub.2, 78.sub.3, (FIG. 2C). The control processor 10.sub.d can use the I/O buffer 36 (FIG. 1A) to communicate to flag register 24, the interrupt register 34 and the mask register 36.sub.m in order to flag errors or to interrupt data transfers on a given protocol channel bus 6 or to mask out the use of the integrity checking system when data is being transferred without an EDC signature.

Detailed Description Text (13):

As seen in FIG. 1B, there may be multiple numbers of protocol controller busses (6.sub.0, 6.sub.1, 6.sub.2, 6.sub.3) connected to multiple numbers of sending/receiving modules (80-24.sub.d). Each time a selected set of these sending/receiving sets are used, the counter 32 is set to indicate the number of bytes in a data block to be transferred. When after this data transmission, the counter reaches zero to indicate completion, then the control-processor 10.sub.d will choose another set of sender-receiver bus units for data transfers at which time the counter will be reset to a new number to indicate the bytes and the data blocks to be transferred.

Detailed Description Text (14):
When all the available data is transferred, the counter feeder controller 32 will indicate a zero via flip-flop 48, FIG. 1A, and then, it will remove the bus grant from the SPC 80 to stop further usage because there is no more valid data available in the buffer 24.sub.d. At the same time, counter feeder controller 32, FIG. 1A, will set 1 bit in the interrupt register 34 for the controller processor 10.sub.d to determine the source of the interrupt at a later time, and it (32) will interrupt the control processor 10.sub.d.

Detailed Description Text (15):
Thus, with this notification, the DIM control processor 10.sub.d can then determine when the memory buffer 24.sub.d must be supplied with more data so that more data is available for data transfer operations. Likewise, the DIM processor 10.sub.d can also then focus on selecting another one of the multiple protocol channel busses for data transfer operations between another set of sending-receiving modules.

Detailed Description Text (17):
The device interface module 8.sub.d keeps an identification or ID register to determine, at any given time, as to which device or sending-receiving set in the system is actively processing the transfer of data. A pointer is held in the SCSI Control Processor 80 regarding data in the memory buffer 24.sub.d. If an identification number ID found by the processor controller 10.sub.d matches with the given protocol-controller SPC 80 that the control-processor wants to update, the processor 10.sub.d reads the value from the SPC pointer and calculates the new value of the pointer based on the amount of data that has accumulated in the main memory buffer 24.sub.d. Then, the SPC pointer is updated and the channel bus 6 is released for use by the protocol controller SPC 80.

Detailed Description Text (19):
Referring to FIG. 1B, each channel bus 6.sub.0, 6.sub.1, 6.sub.2, 6.sub.3, is seen to have a Data Feeder Control (DFC) Circuit F.sub.0, F.sub.1, F.sub.2, F.sub.3) is which abstractly shown with a counter (32.sub.0, 32.sub.1, 32.sub.2, 32.sub.3 which is preset with the number of blocks of data to be transferred between a SPC (80.sub.0, 80.sub.1, 80.sub.2, 80.sub.3) and a dedicated segment in buffer memory 24.sub.d. Additionally, each DFC circuit is indicated to have an interrupt register (34.sub.0, 34.sub.1, 34.sub.2, 34.sub.3) which will send an interrupt to control processor 10.sub.d when, for example, the I/O transfer cycle is completed or if there was an EDC error.

Detailed Description Text (20):
The bus arbitration logic 10.sub.a, seen in FIG. 1A, will insure that each channel bus 6 gets equal time periods of access for data transfer operations. Alternatively, the control processor can be programmed to give access to a requesting SPC 80 when other of the SPCs are idle.

Detailed Description Text (23):
The initialize register 22 will also indicate the "size" of the block of data being transferred. The input register 14 makes a copy of each data word in the entire block. The input register 14 looks at each block and creates a internally generated signature. The purpose of this is to compare the original EDC signature provided at the end of each block (FIG. 4) with the locally internally generated EDC signature by register 14.

Detailed Description Text (24):
The counter 26 is used to count the number of data words transferred between the two digital modules 24.sub.d and 80. The SCSI Protocol Controller, SPC 80, loads the "size" of the data

block into the initialize register 22 before the data transfer. When the counter 26 reaches its set limit, the data in the initialize register 22 is loaded into the block counter 26 to indicate the block size for the "next" block transfer on the present channel bus or on another channel bus 6.

Detailed Description Text (25):
The Protocol Processor, SPC 80, can read the initialization register 22 in order to save its content when it switches to data transfers of a different block size. Additionally, the SPC 80 can restore the old value of the block counter 26 by writing into these registers.

Detailed Description Text (26):
The initialization register 22 is programmable by the SPC 80 dynamically. The SPC 80 has a processor which loads the data block size into the initialization register 22 before the data transfer. When the block counter 26 reaches its set limit, then the initialization register 22 data on block size is loaded into the block counter 26 for the next block transfer. This is done when the zero detector circuit 42 senses when the counter 26 has reached its limit (zero) and thus permits a reloading of the counter.

Detailed Description Text (31):
The EDC generator 28 generates an internal EDC signature for the data being transferred which is then compared to the original EDC signature sent along with the originally transmitted data. The output of generator 28 is placed in the EDC register 40. From this, the error detection circuit 46 can detect whether there has been an EDC error detected or not. If so, then the flip-flop 46 is initiated in order to provide a EDC error interrupt signal on line 46.sub.e.

Detailed Description Text (35):
The sequence of operation when data is being transferred from the memory buffer 24.sub.d over to the SCSI Protocol Controller 80 will operate as follows. The initialize register 22 will receive data to sense the size of the block (a block of "X" bytes or a block of "Y" bytes).

Detailed Description Text (45):
The zero detector circuit 21 (FIG. 5) indicates when the 512 bytes of data have been transferred between digital modules, i.e. the end of the data block transfer.

Detailed Description Text (51):
For example, if the main host processor 4 commands that 100 blocks of data be transferred to the memory buffer 24.sub.d and then from the memory buffer 24.sub.d to a particularly designated peripheral unit 70, it should be understood that at certain times the buffer memory 24.sub.d may not have 100 blocks of data in it but may have only 10 blocks or 20 blocks of data in it. Under these conditions, the DFC counter (32, FIG. 1A) monitors the situation so that if the buffer memory 24.sub.d only has 10 blocks, then only 10 blocks will be permitted to be transferred from the buffer memory 24.sub.d to the SPC 80 and then to the peripheral unit 70.

Detailed Description Text (52):
When adequate data has been transferred from the host processor 4 and the channel interface module 8.sub.c to the memory buffer 24.sub.d, then the DFC counter 32 will permit data to be transferred on the protocol channel bus 6. So even though a particular SPC 80 has been commanded to transfer 100 blocks, it will only be permitted to transfer the number of blocks that are "presently residing" in the memory buffer 24.sub.d. The DFC Bus Arbitration Logic 10.sub.a with its counter 32 will deny any further bus grants to a protocol channel bus 6 when all the available data in the buffer memory 24.sub.a has been transferred.

Detailed Description Text (54):
Step 1: Assuming there are 10 blocks in the buffer memory 24 and the SPC 80 is set to transfer 100 blocks as commanded by host 4. However, the counter 32 in the data feed controller (DFC) is set for 10 blocks only since that is the amount presently available in the memory buffer 24.sub.d. At this point the SPC 80 can start the transfer of data.

Detailed Description Text (55):
Step 2: During the period of earlier transfer of 10 blocks of data, the buffer has now received

10 more blocks of data from the main host computer 4. Here the controller-processor 10.sub.d updates the counter 32 in the data feed controller (DFC) with the additional 10 blocks. Now the SPC 80 can continue to transfer the additional 10 blocks.

Detailed Description Text (58):
These type of operations can continue for each of the four channels involving the four SPC 80 protocol controllers (80.sub.0, 80.sub.1, 80.sub.2, 80.sub.3). The identification number (ID) sector in the memory buffer 24.sub.d will keep track of the data being transferred on each channel. In each case, since there is only one counter 32 available for each set of sender-receiver modules, the DFC control counter 32 will be updated to reflect the count figure of data blocks to be transferred for each one of the channels as they are activated.

Detailed Description Text (59):
Described herein has been a data feeder control system which operates to control and regulate the amount of data between a multiplicity of sending-receiving modules which transfer data on a plurality of protocol channel busses. The data transferred on the protocol channel bus is checked for integrity in two ways. One way is a parity check for each word and the other is via an error detection code (EDC) for each block transferred.

Detailed Description Text (62):
During the occasions of multiple channel operations occurring on different sets of sender-receiver units, the termination of data transfers in a given set will indicate (through a flag register) to the processor controller 10.sub.d that a certain number of data blocks have been completed on a certain set of sender-receiver units. This allows another set of sender-receiver units to be chosen for data transfer operations. Additionally, the processor controller 10.sub.d can also enable more data to be transferred into or out of the memory buffer 24.sub.d depending on what the requirements are at any given time.

CLAIMS:

1. A data feeder and management system using a processor-controller means for controlling data transfers between a plurality of main host computers and a plurality of peripheral devices, said data composed of "B" blocks of "d" bytes which include an original error detection code (EDC) signature for each block, said system comprising:

(a) a memory buffer means for holding blocks of data being transferred between a main host computer and a specified peripheral device, said buffer means connected to said main host computers and including (i) dedicated segments for holding data being transferred to/from N peripheral devices on N channel bus means where each segment includes ID means for recording the amount "B" of data blocks to be transferred on each one of said N channel bus means; and (ii) means to transfer said data block amount of "d" bytes to an associated protocol controller means;

(b) a plurality N of said protocol controller means, each protocol controller means having a channel bus means connected to said memory buffer means via a selecting multiplexer means, each said protocol controller means connecting to a peripheral device which can receive or send data;

(c) said processor-controller means for tracking the number of data blocks transferred on each one of said N channel bus means, said processor-controller means including channel bus arbitration means for enabling said selecting multiplexer means to select one of said channel bus means for data transfer;

(d) a plurality of N Data Feeder Control means, each Feeder Control means attached to an associated channel bus means and including:

(d1) counter means for registering when the number of data blocks transferred has reached the amount of "B";

(d2) interrupt means to remote accessibility to the associated channel bus means which has

successfully transferred "B" blocks of data by signalling said processor-controller means;

(e) a plurality of N integrity checking circuit means, operating directly on-the-fly during data transfers on said channel bus means, each said integrity circuit means attached to one of said channel bus means and including:

(e1) means to parity check each byte of data transferred;

(e2) means to generate a resultant error detection code (EDC) signature for each block of data transferred;

(e3) means to compare said original EDC signature with said resultant EDC signature, and to generate an error signal if a mismatch occurs.

2. The system of claim 1 wherein each of said protocol-controller means includes means to receive a command from said main host computer to transfer "B" blocks of data and store said number "B" until "B" blocks have been transferred.

6. A data feeder control and management system for the monitoring and supervision of multiple numbers of sender-receiver modules wherein a sender module transmits data organized in "B" blocks having "d" bytes including a header indicating the number "d" of bytes in each block and an original error detection code (EDC) for each block, each set of sender-receiver modules connected by a channel bus means, said system comprising:

(a) a multiple number of N sets of sender-receiver modules, each set connected by its own channel bus means;

(b) N integrity checking means operating on-the-fly for checking the parity of each byte of each word transferred on each of said N channel bus means and for checking the original error detection code (EDC) signature of each block transferred on each of said channel bus means, said checking means including:

(b1) means to generate a resultant EDC for each block transferred for comparison with said original EDC block;

(b2) means to signal an error signal to said processor-controller means when said EDC codes do not match.

(c) processor controller means connected to each of said channel bus means for regulating data transfers on each channel bus means, including:

(c1) means to select any one of said multiple sets of sender-receiver units to set the prescribed amount of data to be transferred and to signal said processor-controller means when the prescribed amount of data has been transferred;

(d) N data feeder control means, each means including: counter controller means to register the amount of data transferred on each one of said sets of sender-receiver modules and to notify said processor-controller means when the prescribed amount of data has been transferred;

(e) means to interrupt and inform said processor-controller means when said prescribed amount of data transfer has been completed.

Previous Doc Next Doc Go to Doc#